Google™

Fastfood

O(n log d) feature maps for kernels

Tamas Sarlos, Quoc Le, Alex Smola

The trouble with kernels

- Kernel expansion $$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x)$$
- Number of basis functions increases linearly with sample size
  Inevitable unless you have essentially noise free cases (Steinwart & Christmann)
- Approximate expansions are slow (d dimensions, n features, m samples)

	CPU Training	CPU Test	RAM Training	RAM Test
Naive	$O(m^2 d)$	$O(md)$	$O(md)$	$O(md)$
Reduced set	$O(m^2 d)$	$O(nd)$	$O(md)$	$O(nd)$
Low rank	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$
Random Kitchen Sinks	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$

The trouble with kernels

- Kernel expansion $$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x)$$
- Number of basis functions increases linearly with sample size
  Inevitable unless you have essentially noise free cases (Steinwart & Christmann)
- Approximate expansions are slow (d dimensions, n features, m samples)

	CPU Training	CPU Test	RAM Training	RAM Test
Naive	$O(m^2 d)$	$O(md)$	$O(md)$	$O(md)$
Reduced set	$O(m^2 d)$	$O(nd)$	$O(md)$	$O(nd)$
Low rank	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$
Random Kitchen Sinks	$O(mnd)$	$O(nd)$	$O(nd)$	$O(nd)$
Fastfood	$O(mn \log d)$	$O(n \log d)$	$O(n)$	$O(n)$

Random Kitchen Sinks (Rahimi & Recht,

- RBF kernels have Fourier representation (Bochner, 1932)

$$k(x, x') = \int dp(\omega) e^{i\langle \omega, x \rangle} e^{-i\langle \omega, x' \rangle}$$

- Draw frequencies from p(w) and approximate kernel

$$k(x, x') = \frac{1}{n} \sum_{j=1}^{n} e^{i\langle \omega_j, x \rangle} e^{-i\langle \omega_j, x' \rangle} \text{ where } \omega_j \sim p(\omega)$$

  - Special case – for Gaussian RBFs we draw terms from a Gaussian (Fourier transform of Gaussian)
  - In general, draw w from spherically symmetric distribution (e.g. Gauss) and rescale

- Dominant cost

$$\Omega x \text{ is } O(nd) \text{ CPU and } O(nd) \text{ RAM}$$

- Can we accelerate this? Does this work for other problems, too?

Key Idea

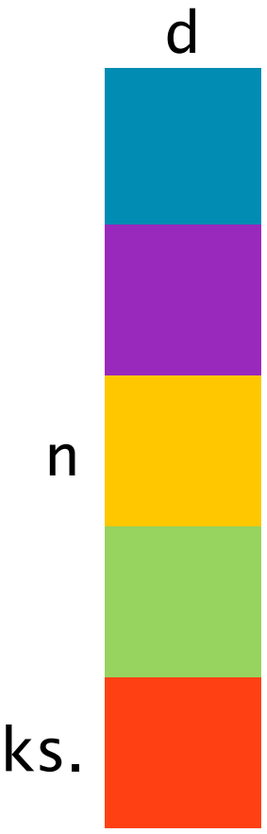
- Gaussian matrix M costs O(n d) per multiplication
- Approximate by ‘fake’ Gaussian matrix (for moment assume that M is square)

$$\tilde{M} = SHG\Pi HB$$

  - S is random diagonal scaling matrix (deals with spectrum)
  - H is Hadamard matrix admitting O(d log d) multiply

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} \text{ and } H_1 = 1$$

  - G is random diagonal Gaussian matrix
  - Π is random permutation matrix
  - B is random binary {-1, 1} diagonal matrix
- Multiplication is O(d log d). Storage is O(d). Draw independent blocks.



Properties of $\tilde{M} = SHG\Pi HB$

- Correct expectation
  - Ignore S – now each row is drawn from iid Gaussians
  - Unfortunately all rows have same length, given by $\|G\|_{\text{Frob}}^2$
  - Use S to randomize lengths (and adapt to different spectra)
- Covariance between features is well controlled

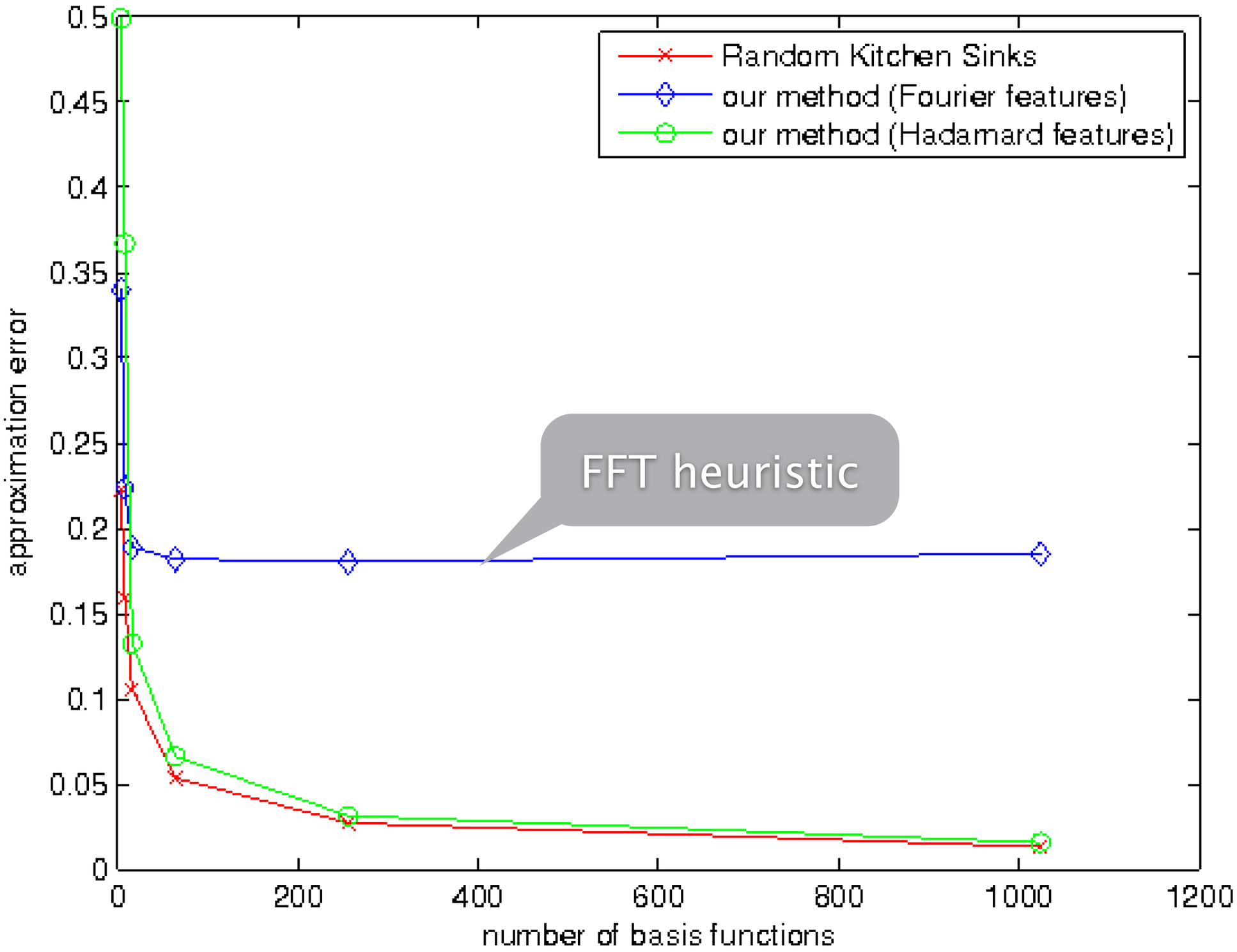
Theorem 4 (Low Variance) *Let $v = x - x'$ and denote by $\psi_j(v) = \cos(d^{-\frac{1}{2}}[HG\Pi HBv]_j)$ the jth random feature for $j \in \{1 \dots d\}$. Then for each j we have*

$$\text{Var}\left[\psi_j(v)\right] = \frac{1}{2}\left(1 - e^{-\|v\|^2}\right)^2 \tag{16}$$

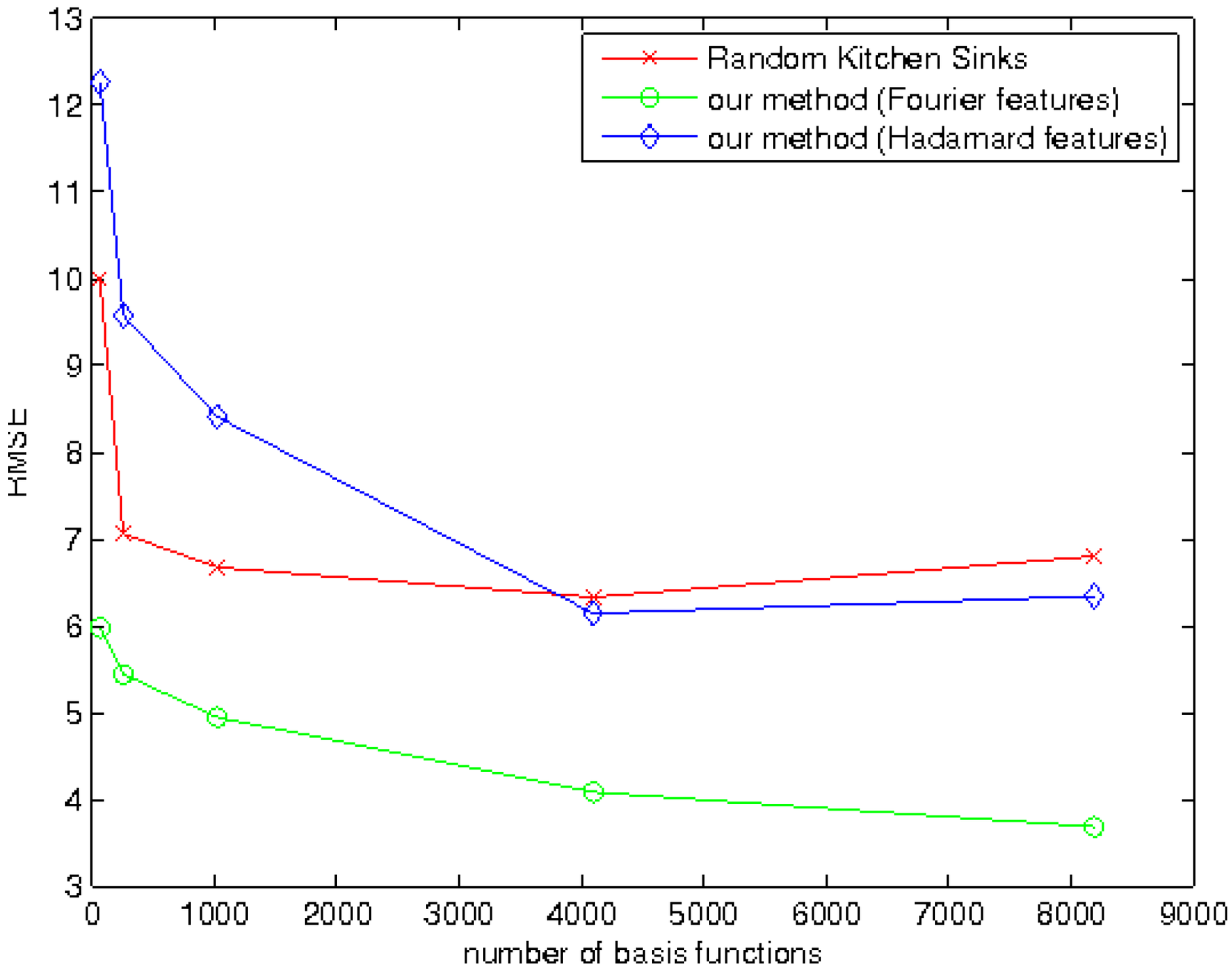
$$\text{Var}\left[\sum_{j=1}^{d} \psi_j(v)\right] \leq \frac{d}{2}\left(1 - e^{-\|v\|^2}\right)^2 + dC(\|v\|) \tag{17}$$

where $C(\alpha) = 12\alpha^4\left[e^{-\alpha^2} + \frac{\alpha^2}{3}\right]$.

Matrix approximation error



Generalization Performance (UCI CPU



Speed & accuracy

Dataset	m	d	Exact	Nystrom	Random Kitchen Sinks	Fastfood FFT	Fastfood
Insurance Company (COIL2000)	5,822	85	0.231	0.232	0.266	0.266	0.264
Wine Quality	4,080	11	0.819	0.797	0.740	0.721	0.740
Parkinson Telemonitor	4,700	21	0.059	0.058	0.054	0.052	0.054
CPU	6,554	21	7.271	6.758	7.103	4.544	7.366
Relative location of CT slices (axial)	42,800	384	n.a.	60.683	49.491	58.425	43.858
KEGG Metabolic Reaction Network	51,686	27	n.a.	17.872	17.837	17.826	17.818
Year Prediction MSD	463,715	90	n.a.	0.113	0.123	0.106	0.115
Forest	522,910	54	n.a.	0.837	0.840	0.838	0.840

works fine
(no difference)

much faster

d	n	Fastfood	RKS	Speedup	RAM
1,024	16,384	0.00058s	0.0139s	24x	256x
4,096	32,768	0.00136s	0.1224s	90x	1024x
8,192	65,536	0.00268s	0.5360s	200x	2048x

Summary

- Extensible to other kernels
  - Easy to sample spectral distribution
  - Easy to learn ‘multiple kernels’ since dimensions are so cheap
- Extensible to localized basis functions
  - Instantiate Neal’s 1994 paper (GP and Neural Network equivalence)
  - Matrix valued functions
- Iterate
  - Stack several layers
  - Backprop is very cheap since inverse Hadamard is O(d log d)
- k(x,x’) is now expensive (invariance theorem of difficulty …)
- Covariance operators are not explicitly available
- Never store explicit feature map (too much memory required)